2

DESIGN PRINCIPLES FOR CONNECTED DEVICES

THIS BOOK IS called *Designing the Internet of Things,* so does that mean we think design is an important part of building a connected device?

For some applications, you may think that design isn't important at first glance. Who cares what the box holding a production-line sensor looks like in a factory? Surely form is defined by function?

Although the physical design of your connected device may be less important than that of something you're going to set on your mantelpiece, the extra functionality regarding how it will interact with the rest of the factory and its control systems is something you should consider and think through rather than just leave to chance.

If you haven't spent much time with designers, you might be forgiven for thinking that design is merely concerned with the shape and look of an object—something ornamental, to give it a pleasing appearance. However, design is a much wider field than that.

Industrial design (sometimes also called *product design*) does include the form and decoration of the item, but it also covers functional aspects such as working out how the product will be constructed or ensuring the controls are easily understood.

The user interface to the object—be that on a screen or the more traditional buttons and switches—is also something of interest to the discipline of experience design. That takes the perspective of the end user of the design as its focus and seeks to create the best solution for the user. Obviously "best" is subjective and could aim to make using the device as enjoyable as possible, or perhaps as efficient as possible, depending on the priorities of the designer or her team.

The rise of digital services, particularly those which take advantage of the Internet and the network effects it enables, has led to design specialists who take a wider view of the design of the whole system. Service design has the broadest view of the service in its entirety, whereas interaction design also looks at how different parts of the system interrelate, and especially how the user features in that interaction.

There are no hard boundaries between these differing facets of design, but we think designers of all types would agree that design is about more than just the surface look of the device.

In this chapter we look at some of the overarching principles that can be applied when designing an Internet of Things system, and then visit some of the techniques you can use to help explore the problem space and end up with a stronger product. Not all the techniques apply in all cases, but they provide some useful rules of thumb in approaching your work.

CALM AND AMBIENT TECHNOLOGY

The Internet of Things has its roots in the work done by Mark Weiser at Xerox PARC in the 1990s. His work didn't assume that there would be network connectivity but was concerned with what happens when computing power becomes cheap enough that it can be embedded into all manner of everyday objects. He coined the term *ubiquitous computing*, or *ubicomp* for short, to describe it, and through his research and writing sought to explore what that would mean for the people living in such a world.

With its focus on computing power being embedded everywhere, ubicomp is often also referred to as *ambient computing*. However, the term "ambient" also has connotations of being merely in the background, not something to

which we actively pay attention and in some cases as something which we seek to remove (e.g., ambient noise in a sound recording).

We prefer, as did Mark Weiser, the term *calm technology*—systems which don't vie for attention yet are ready to provide utility or useful information when we decide to give them some attention.

Such proliferation of computing devices into the world comes with all manner of new challenges. Issues include configuration, how to provide power to all these items, how they talk to each other, and how they communicate with us.

The power and networking challenges are purely technical and are driving developments such as 6LoWPAN (`www.ietf.org/dyn/wg/charter/6lowpan-charter.html`). This is a standards drive from a working group of academics, computing professionals, and others to take the next-generation Internet protocol (IPv6) to the simplest and lowest-power networked sensors. (It is revisited when we look at future developments in the next chapter.) It aims to provide the scale of addresses and lower power usage needed by so many sensors.

Configuration and user interaction, however, obviously involve people and so are difficult problems to solve with just technical solutions. This is where good design can aid in adoption and usability. You can see this with the introduction of the Apple iPod in 2001. It wasn't the first portable MP3 player, but the combination of the scroll-wheel user interface and the companion iTunes software made it much easier to use and turned them into mass market gadgets.

Designing a connected device in isolation is likely to lead you to design decisions which aren't ideal when that object or service is placed into the seething mess that is the real world. To bastardize Eliel Saarinen's maxim on design, we suggest you think of how the connected device will interact as one of a wealth of connected devices.

In addition to thinking of a device in the physical context one step larger—Saarinen's "Always design a thing by considering it in its next larger context—a chair in a room, a room in a house, a house in an environment, an environment in a city plan"—we should do the same for the services.

For connected devices which are just sensing their world, or generally acting as *inputs*, as long as their activity doesn't require them to query the people around them, there shouldn't be any issues. They will happily collect information and deposit it into some repository online for processing or analysis.

When the devices start interacting with people, things get more complicated. Already we're seeing the number of notifications, pop-ups, and indicator noises on our computers and mobile phones proliferate. When we scale up this number to include hundreds of new services and applications and then spread that across the rest of the objects in our world, it will become an attention-seeking cacophony.

Mark Weiser and John Seely Brown proposed an antidote to such a problem by suggesting we design ubiquitous computing systems to seek to blend into their surroundings; in so doing, we could keep them in our peripheral perception until the right time to take centre stage:

> *Calm technology engages both the center and the periphery of our attention, and in fact moves back and forth between the two.*
>
> *—Designing Calm Technology, Mark Weiser and John Seely Brown, Xerox PARC, December 21, 1995*

A great example of this approach is *Live Wire*, one of the first Internet of Things devices. Created by artist Natalie Jeremijenko when she was in residence at Xerox PARC under the guidance of Mark Weiser, Live Wire (also sometimes called Dangling String) is a simple device: an electric motor connected to an eight-foot long piece of plastic string. The power for the motor is provided by the data transmissions on the Ethernet network to which it is connected, so it twitches whenever a packet of information is sent across the network.

Under normal, light network load, the string twitches occasionally. If the network is overloaded, the string whirls madly, accompanied by a distinctive noise from the motor's activity. Conversely, if no network activity is occurring, an unusual stillness comes over the string. Both extremes of activity therefore alert the nearby human (who is used to the normal behaviour) that something is amiss and lets him investigate further.

> *Not all technology need be calm. A calm videogame would get little use; the point is to be excited. But too much design focuses on the object itself and its surface features without regard for context. We must learn to design for the periphery so that we can most fully command technology without being dominated by it.*
>
> *—Designing Calm Technology, Mark Weiser and John Seely Brown, Xerox PARC December 21, 1995*

The mention of the distinctive sound from the motor when the Live Wire is under heavy load brings up another interesting point. Moving the means of conveying information away from screens and into the real world often adds a new dimension to the notification. On a computer, updating the screen is purely visual, so any additional senses must be engaged explicitly. Like Live Wire, Bubblino—Adrian's Internet of Things bubble machine which searches Twitter and blows bubbles when it finds new tweets matching a search phrase (see the case study in Chapter 4)—is a good example in which the side effect of the motor is to generate an audible notification that something is happening. With their Olly (`www.ollyfactory.com`) device, agency Mint Digital combines the motor with a deliberate olfactory indicator to provide a smelly notification of one of a number of social media events.

These noisy "side effects" are something that we should also be wary of losing with a move to "better" technology. Years ago all airport and railway arrival and departure boards were built using split-flap displays. They consisted of a number of flaps on a roll—sometimes with full place names printed onto the flap, and in other times as individually controllable characters—which could be rotated until they showed the correct item.

In most locations these split-flap displays have been phased out in preference for dot-matrix LED displays. The newer displays are much easier to update with new destinations. They also have capabilities such as horizontally scrolling messages which were impossible to add with the split-flap technology. Sadly, in doing so they have lost one important characteristic: the flurry of clacking as the display updates. As a result, passengers waiting in a station terminal must stare endlessly up at the display waiting for their train to be announced, rather than attending to other tasks and checking the departures board only when a change occurs.

That is not to say that screens are never the right choice, merely that in this age of mobile phones and tablets they are often chosen without realising a choice is being made. If you start from a position of trying *not* to use a screen, then if you return to it you will have worked out that a screen is the best solution.

There has been some interesting experimentation in the use of screens around what has been called *glanceable displays*. These are secondary screens, meant to sit away from your immediate surroundings in the same sort of places in which you might place a picture frame.

They aren't all screens. For example, Russell Davies, agitator for the recently possible, built Bikemap (`http://russelldavies.typepad.com/planning/2011/04/homesense-bikemap.html`), a handful of LEDs

inserted into specific places on a printed-out map. The map shows the area around his home, and each LED marks the location of a bike stand for the London city bike rental scheme. If there are more than five bikes available at a stand, the corresponding LED lights up. It is mounted into a picture frame and hangs near to Russell's front door, so a glance over to it as he leaves lets him know which direction to head in order to find a bike.

One of Russell's roles is as a partner in the Really Interesting Group, a multidisciplinary agency based in London. Others in the agency, and some of their wider network of friends, have also been exploring the area.

They have a set of AirTunes WiFi speakers in the studio, which anyone can take control of and play music through. When you were working there, you'd often wonder exactly what a particular track was but had no way of finding out short of interrupting the entire office to ask who was in charge of the music at that moment and what was playing right now.

To solve that problem, they stuck a spare monitor out of everyone's way on top of a bookcase and, through a combination of watching the network traffic and hooking into the last.fm service that they all used to record what tracks they play, built a system to display the current track and who had played it.

The screen updated only whenever the song changed, and wasn't positioned in anyone's eye-line, so it didn't distract you from your work. However, if the music distracted you, the screen was there to satisfy your curiousity.

The Bikemap also provided some inspiration for RIG studio-mate Chris Heathcote. Chris is an interaction designer and realised that every morning he would check a few different apps on his phone to find out things like the weather forecast, his appointments for the day, and how the trains on the London Underground were running.He didn't have any power sockets near to his front door, and so settled on a bedside information display instead. Given that it would be always on and next to where he sleeps, a standard monitor or other LCD display, with its persistent glow, wouldn't be suitable. The e-ink display on a Kindle, however, was ideal. He took advantage of the WiFi connectivity and computing power in the Kindle to make it a self-contained device and configured it to just display a web page, which refreshed every few minutes.

The resultant device, which Chris called the Kindleframe (`http://anti-mega.com/antimega/2013/05/05/kindleframe`), would then always display up-to-date information from the mash-up of websites that he pulled together to collect all the information that he needs at the start of the day.

MAGIC AS METAPHOR

One of the main issues with introducing any new technology or service that is radically different from the norm is getting people to understand and accept it. Early adopters are generally happier looking a bit strange or doing things somewhat awkwardly to reap the benefits of the new gadgets; however, for the technology/service to catch on, you need to persuade the majority to take it up.

In addition to the technology becoming capable of a particular action, we often need *society*, for wont of a better term, to be ready to accept it. There are many examples when the main difference between a failed technology and a wildly successful one is that the successful one arrived a few years later, when people were more receptive to what was offered.

Technology blogger Venkatesh Rao came up with a good term to help explain how new technology becomes adopted. He posits that we don't see the present, the world that we live in now, as something that is changing. If we step back for a second, we do *know* that it has changed, although the big advances sneak up on us over time, hidden in plain sight. Rao called this concept the *manufactured normalcy field* (`www.ribbonfarm.com/2012/05/09/welcome-to-the-future-nauseous/`).

For a technology to be adopted, it has to make its way inside the manufactured normalcy field. As a result, the successful user-experience designer is the one who presents users with an experience which doesn't stretch the boundaries of their particular normalcy field too far, even if the underlying technology being employed is a huge leap ahead of the norm. For example, the mobile phone was first introduced as a phone that wasn't tethered to a particular location. Now broadly the same technology is used to provide a portable Internet terminal, which can play movies, carry your entire music collection, and (every now and then) make phone calls.

The way that portable Internet terminals made it into our manufactured normalcy field was through the phone metaphor. Introducing technology to people in terms of something they already understand is a tried and tested effect: computers started off as glorified typewriters; graphical user interfaces as desktops....

So, what of the Internet of Things? As we saw in the last chapter, Arthur C. Clarke has claimed that "any sufficiently advanced technology is indistinguishable from magic," and given that the Internet of Things commonly bestows semi-hidden capabilities onto everyday objects, maybe the enchanted objects of magic and fairy tale are a good metaphor to help people grasp the possibilities.

Some Internet of Things projects draw their inspiration directly from magic. For example, John McKerrell's WhereDial takes its lead from the clock in *Harry Potter* which tracked the location of the members of the Weasley family. The Weasley clock could use magic to divine the whereabouts of each family member and was therefore also aware of when they were in mortal peril. The WhereDial, by comparison, has to rely on mere technology for its capabilities; however, with the GPS chipsets in smartphones and location check-in services like FourSquare, it isn't much of a leap to also own an ornament which updates to show when you are at work, or travelling, or at a restaurant.

CASE STUDY: The WhereDial

Developer John McKerrell is passionate about maps and location. He got a job with mapping startup Multimap when he showed them a version of the slippy maps you could drag around in your web browser (something Google had only just wowed the world with at the time) using the Multimap map tiles, and by 2009 already had a couple of years of regularly logging his location on MapMe.At, a service he built to let people store and share their location.

So, when he attended a physical computing hackday in his home town of Liverpool (coincidentally organised by Adrian), a location-related project was almost a foregone conclusion.

The Harry Potter franchise was near its peak, and in it one of the families, the Weasleys, has a clock which shows where each family member is. A friend suggested that John could use that as a way to visualise some of the location data that he was already gathering.

John sourced a traditional carriage clock and rigged up a stepper motor to drive the clock mechanism, controlled by an Arduino which could talk to his MapMe.At service. The numbers on the clock face were replaced with a collection of common locations: home, work, shops, pub, restaurant, etc. Or, if the person wasn't in a specific place, it would show that they were travelling. That gave him a nice piece to sit on his sideboard and show whomever was at home the current location of both John and his wife (one hand on the clock for each).

Over the following years, John has refined the concept and turned it into an Internet of Things product. The software has been updated to hook into more location services, such as FourSquare, as their popularity has increased, and the physical form of the device itself has also evolved.

Using an existing clock for the housing and face had its issues: there's the challenge of sourcing suitable clocks in sufficient quantities, but also the problem of one of the hands only changing to a new location when the other has done a full rotation (good for showing hours and minutes, less useful for the location of two different people).

The WhereDial.

Now the WhereDial (`http://wheredial.com`) only shows the location of one person, and it is the location display that moves rather than a hand pointing to a location. This revised design is completely bespoke, and John can produce as many as he needs in house using the laser cutter in his workshop.

Other projects have a less obvious influence.

Enchanted mirrors seem a popular choice in design research, although they haven't quite reached the capabilities of the evil queen's "Mirror, mirror on the wall" in the Snow White tale. They tend to show information which is useful as you start or end your day, in line with the expected times when you'd use a bathroom mirror. You get the time and can check appointments and traffic and weather information while having your morning shower. Presumably, it is merely a matter of time before one shows the number of "Likes" you have on Facebook, thus turning it into the modern equivalent of the evil queen's query to know "who is the fairest of them all?"

Given David Rose's thinking around enchanted objects, it's not surprising that Ambient Devices, the company that he used to run, has a couple of examples. The ambient orb is a "single-pixel display" that can show the status of a metric of its user's choosing—the price of a stock, the weather forecast, the pollen count. Like the crystal ball whose shape the orb mimics, it shows you information from afar.

Ambient Devices then took the idea one step further and built an enchanted umbrella. It can read the weather forecast, and the handle glows gently if rain is expected, alerting you to the fact that you may need to pick it up as you head out of the house.

The magic of these devices isn't the epic magic of *The Lord of the Rings*; it's a more mundane, everyday sort of magic that makes tasks a bit easier and lives a little more fun. But that's the point: using our understanding of magic and fairy tales to help make sense of these strange new gadgets.

Of course, tales like *The Sorcerer's Apprentice* show us the danger of trying to use magic to reach beyond our capabilities. When the apprentice tries to ease his chores by using magic, that he doesn't fully understand, to enchant the broom, things get out of hand. It takes the timely return of the sorcerer to restore order. So far, our Roomba automated vacuum cleaners seem relatively well behaved, but we should be wary of creating Internet of Things devices whose seamless "magical" abilities give them behaviours or control interfaces which are hard for their owners to comprehend.

As well as trusting that your devices will do your bidding, it is also important to trust them to safeguard any data that they gather.

PRIVACY

The Internet of Things devices that we own aren't the only ones that should concern us when it comes to matters of trust. With more sensors and devices watching us and reporting data to the Internet, the privacy of third parties who cross our sensors' paths (either by accident or design) is an important consideration. Designers of an Internet of Things service will need to balance these concerns carefully.

KEEPING SECRETS

For certain realms, such as health care, privacy concerns are an obvious issue, so we cover this topic in more detail in Chapter 11, "Ethics". However, even seemingly innocuous applications can leak personal information, so you should be alert to the danger and take measures to avoid it.

This advice is perfectly illustrated with an example from an early instrumented car park in a Westfield shopping mall in Australia. Each parking bay is overlooked by a small sensor from Park Assist, which uses a cheap camera to tell whether the space is occupied. The sensors are all networked and presumably can provide analytics to the owner of the car park as to its usage. A light on the sensor can help guide drivers to a free space. All useful and harmless stuff.

The problem came with a more advanced feature of the system. The shopping mall provided a smartphone app for visitors to download so that they could find out more information about the facilities. One of the features of the app was a Find My Car option. Choosing that, you were prompted to enter the first few characters of your licence plate, and the app would then return four small photos of potential matches—from optical character recognition software processing the sensor data on the mall's server.

The returned images were only thumbnails—good enough to recognise which was your car, but not much else, and the licence plates were blurry and hard to see. However, security professional Troy Hunt found that the implementation method left a lot to be desired (`www.troyhunt.com/2011/09/find-my-car-find-your-car-find.html`).

With a fairly simple, off-the-shelf bit of software, Troy was able to watch what information the app was requesting from the server and found that it was a simple unencrypted web request. The initial request URL had a number of parameters, including the search string, but also including information such as the number of results to return.

That request returned a chunk of data (in the easily interpreted, industry standard JSON format), which included the URLs for the four images to download, but also included a raft of additional pieces of information. Presumably, it was easier for the developer of the web service to just return all the available data than to restrict it to just what was needed in this case. The extra data included, for example, the IP addresses of each of the sensor units, but more importantly, it also included the full licence plate for each vehicle and the length of time it had been parked in the space.

By altering the search parameters, Troy found that he could request many more than the four matches, and it was also possible to omit the licence plate search string. That meant he could download a full list of licence plates from all 2550 parking spaces in a single web request, whenever he liked.

Obviously, all that data is already publicly available, but there's a pretty large difference in ease of gathering it between staking out the entrance to the car park and watching cars come and go and setting up a script on a computer to check at regular intervals.

Once alerted to the problem, Westfield and Park Assist were quick to disable the feature and then work with Troy to build a better solution. However, that situation came about only because Troy was generous enough to bring it to their attention.

Don't share more than you need to provide the service.

As founder of WikiLeaks, Julian Assange, has said, "The best way to keep a secret is to never have it" (`www.pbs.org/wgbh/pages/frontline/wikileaks/interviews/julian-assange.html`). If you can avoid gathering and/or storing the data in the first place, you need not worry about disclosing it accidentally.

In this day and age, it is standard practice to never store passwords as cleartext. You could also consider applying the standard mechanisms for password encryption, such as the one-way hash, to other pieces of data. This technique was suggested by Peter Wayner in his book *Translucent Databases* (CreateSpace Independent Publishing Platform, 2009). Rather than storing identifying data in the database, if you don't need to return it to its original form (that is, you just need it to be unique and associated with the same group of data), use a one-way hashed version of the information instead. Doing so still lets the originators of the data find their data (as they can provide it to be hashed again) and allows statistics gathering and reports, and the like, without storing the data in a recoverable form.

Hashes

One-way hashing is a cryptographic technique used to condense an arbitrarily sized chunk of data into a fixed-sized piece, called the hash. It's called one-way hashing because there isn't an easy way, given the resultant hash, to work out what the original data was. Hashing algorithms are so designed such that even a small difference in the input data leads to a huge difference in the output hash.

This makes them very useful for times when you want to verify that two pieces of data are identical without having to store them for comparison. That's useful when the data you want to compare is either very large or something you don't want to store in its original form.

The most common use of cryptographic hashes is in password verification. Rather than store the user's password, the service provider stores a hash of the password. When the user wants to authenticate himself in the future, the hash can be recalculated; if it matches the stored one, the service can be reasonably sure that the user has provided the correct password.

It is good practice to salt the password before applying the hash. This adds some random, non-secret extra text to the password before the hash is computed. The salt is then stored with the hash, so the service can concatenate the two again when it needs to verify a newly presented password. The salt prevents any attacker who ends up with a copy of the hash from easily comparing it to a dictionary of precompiled hashes to work out the password.

WHOSE DATA IS IT ANYWAY?

With the number of sensors being deployed, it isn't always clear whose data is being gathered. Consider the case of a camera deployed in an advertising hoarding which can check to see whether people are looking at the different adverts. Does the data belong to the company that installed the camera or to the members of the public who are looking at the adverts? Adam Greenfield, a leading practitioner of urban computing, makes a convincing argument that in a public space this data is being generated by the public, so they should at least have equal rights to be aware of, and also have access to, that data. (See point 67 at `https://speedbird.wordpress.com/2012/12/03the-city-is-here-for-you-to-use-100-easy-pieces/`.)

On private property, you can more easily claim that the members of the public don't have such a right, but perhaps the property owner might assert rights to the data rather than whoever installed the camera. And there are many places such as shopping malls which, to all intents and purposes, look and feel like public spaces, despite being privately owned. How do things stand in those areas?

When convening to debate such issues in the summer of 2012, the participants at the Open Internet of Things Assembly (`http://openiotassembly.com/`) coined the term *data subjects*—those people to whom the data pertains, regardless of whether they owned the sensors used to gather the data or the property where the sensors were sited. There's no clear understanding of what rights, if any, such "data subjects" will enjoy, but it is an area that deserves more debate and attention.

WEB THINKING FOR CONNECTED DEVICES

When you are thinking of the networked aspect of Internet of Things objects, it might help to draw on experiences and design guidelines from existing network deployments. The obvious choice would be to look at design guides to the World Wide Web and the Internet itself; after all, the term *Internet of Things* will look quaint in the future when we accept that it is completely natural for the Internet to be full of Things as well as computers and phones. You should aim to get into the mindset of the web and create devices which are *of* the web rather than those which just exist *on* the web.

In an early version of the specification for TCP (RFC761, `http://tools.ietf.org/html/rfc761 - section-2.10`), Jon Postel wrote: "Be conservative in what you do, be liberal in what you accept from others". Since then, that *robustness principle* has become so well known that it is commonly referred to as *Postel's Law*. It is good to bear this in mind when designing or building anything which must interact with other services—particularly when you aren't the one building the other components with which your system interacts.

SMALL PIECES, LOOSELY JOINED

Even if you are building all the components of your service, it makes sense not to couple them too tightly together. The Internet flourished not because it is neatly controlled from a central location, but because it isn't; it is a collection of services and machines following the maxim of *small pieces, loosely joined*.

What this means for the architects of a service is that each piece should be designed to do one thing well and not rely too much on tight integration with the separate components it uses. Strive also to make the components more generalised and able to serve other systems which require a similar function. That will help you, and others, to reuse and repurpose the components to build new capabilities unimagined when the initial system was commissioned.

Where possible, use existing standards and protocols rather than inventing your own. Any loss of elegance or efficiency of code size or electronics will be outweighed by the availability of standard libraries and skills for people to interact with, and build on, your system. For example, when the designers at Twitter implemented the search feature, they chose to include a more machine-readable option in the form of a feed of results in the standard Atom syndication format (`http://tools.ietf.org/html/rfc4287`). I'm fairly certain that they didn't expect it to be consumed by an Arduino, which would then use it to activate a bubble machine to blow bubbles for each new tweet.

Similarly, because Bubblino was made to understand and consume Atom feeds to look for new tweets, it can now be repurposed to monitor anything with such an interface. So new entries in a blog are trivial to monitor, and most web languages have libraries to generate suitable Atom feeds for any other notifications that you would like to visualise as bubbles.

This meant it was trivial for one customer to get their Bubblino to trigger whenever one of their developers submitted some code which failed their automated tests. They just wrote a small web service that took the test results and presented them as an Atom XML feed. Once their Bubblino was configured to use that new service, it would then blow bubbles whenever the tests failed.

FIRST-CLASS CITIZENS ON THE INTERNET

An extension of the concept of loose coupling is to strive to make your devices first-class citizens on the Internet. What do we mean by that? Where possible, you should use the same protocols and conventions that the rest of the Internet uses.

In the early days of any new development, it is tempting to compromise and choose protocols which are easier to implement. Indeed, many middleware providers encourage that practice, claiming that such low-powered end-points are not capable enough (in processing power or RAM or in the networks they use). There is an element of truth to this—though not as much as you will be led to believe—but a good rule of thumb for the past 20 years or more has been to expect the IP protocol to penetrate everywhere. We see no reason for it not to continue into the Internet of Things.

In the few cases where the existing protocols don't work, such as in extremely low-powered sensors, a better solution is to work with your peers to amend existing standards or create new open standards which address the issue within the conventional standards groups.

The evolution of the mobile web serves as a good cautionary example. When mobile phones were first being connected to the Internet, it was deemed too difficult for them to talk to web servers directly, and a whole suite of new protocols, Wireless Application Protocol (WAP), were developed. Handsets accessed either bespoke WAP sites or standard websites via a WAP/web gateway server. As they needed site developers to learn a whole new set of skills, the new protocols gained little traction; without the sites to visit, user adoption of the mobile web was slow.

As the handsets evolved to talk the standard protocols of the web, even though the display wasn't perfect, it was good enough for users to begin using it. With the increase in usage directly to the websites developers could then see the demand for mobile-friendly features. Given that those features could be developed with the tools with which they were already familiar, over time the mobile web has just become a facet of the web in general.

GRACEFUL DEGRADATION

Because the Internet is so welcoming and tolerant of all sorts of devices and services, the endpoints have a massively disparate and diverse range of capabilities. As a result, building services which can be used by all of them is a nearly impossible task. However, a number of design patterns have evolved to mitigate the problem.

The first is to acknowledge that the wealth of different devices is likely to be a problem and design your system to expect it. If you need to come up with a format for some data being transferred between devices, include a way to differentiate between successive versions of the formats—ideally in such a way that older devices can still mostly read newer formats. This is known as *backwards compatibility*, and although over time it will add some cruft to the format, as certain features will only persist to serve outdated devices, it will greatly extend the life and utility of your users' devices. The HTML format does this by stating that any client should ignore any tags (the text inside the <>) that it doesn't understand, so newer versions can add new tags without breaking older parsers. The HTTP protocol uses a slightly different technique in which each end specifies the version of the protocol that it supports, and the other end takes care not to use any of the newer features for that particular session.

The other common technique is to use something called *graceful degradation*. This technique involves aiming to provide a fully featured experience if the client is capable of it but then falling back—potentially in a number of levels—to a less feature-rich experience on less capable clients. This capability can even span technologies and does so in its most common usage. When

trying to implement rich web applications such as Twitter and Gmail, the coder wants to use an assortment of advanced JavaScript features in modern browsers. Well-written apps check that the features are available before using them, but if those features aren't available, the apps might limit themselves to a version using simpler (and more common) JavaScript code—still validating form contents before submitting them, for example, but no longer calling over to the server to provide autocomplete. And if JavaScript isn't available at all, they fall back to basic HTML forms. This experience is not as nice as the full one but better than no experience at all!

Aside from using the same techniques when designing our connected devices, we might also be able to apply that approach to the devices themselves to give a degree of fault tolerance. The proliferation of devices and the likelihood that some of them will break in some way means that it is important that their technology continues to add what value it can, as parts cease to function. When your early-adopter Internet Fridge can no longer talk to your WiFi because it's only IPv4 and the world has moved to IPv6, you would still be able to use its touchscreen to write messages and view the photos stored in the USB stick stuck in it. And if the touchscreen breaks, you should still be able to keep the food inside it cold.

AFFORDANCES

In his book *The Design of Everyday Things*, Donald Norman defines *affordances* as follows:

> *Affordances provide strong clues to the operations of things. Plates are for pushing. Knobs are for turning. Slots are for inserting things into. Balls are for throwing or bouncing. When affordances are taken advantage of, the user knows what to do just by looking: no picture, label, or instruction is required. Complex things may require explanation, but simple things should not. When simple things need pictures, labels, or instructions, the design has failed.*
>
> —The Design of Everyday Things, *MIT Press, 1998*

We recommend this excellent book for anyone with an interest in design, although the section on the affordances of doors may ruin your interactions with buildings because you will encounter examples of poorly thought-out design almost daily.

As adoption of the Internet of Things gathers pace, more and more of our cities, homes, and environment will become suffused with technology. With these additional behaviours and capabilities will come additional

complexity—something that successful designers of connected devices and services will need to counter.

By their very nature, many of the new capabilities bestowed upon objects will be hidden from sight or not immediately apparent from first glance, which makes intuitive design difficult. What are the affordances of digitally enhanced objects?

How do we convey to the user of an object that it can communicate with the cloud? Or that this device is capable of short-range communication such as RFID? What does it mean that a toy knows what the temperature is or when it is shaken? How do you know whether your local bus shelter is watching you or, possibly more importantly, why?

An important start is to keep the existing affordances of the object being enhanced. Users who don't realise that a device has any extra capabilities should still be able to use it as if it hasn't. Although this principle sounds like common sense, it is often discarded due to costs or difficulties in design.

For example, a "dumb" light dimmer switch is usually implemented as a rotary knob which gives the user fine-grained control over the brightness. When it is hooked up to a home-automation system, the difficulties of synchronising the state of both the knob and the light level, now that the brightness can be controlled remotely or automatically, often leads to the knob being replaced with a couple of buttons. As a result, the user loses the ability to make both rapid large changes and smaller, fine-grained adjustments. A better approach would be to adopt the system used on many stereo systems where the volume knob is a motorized potentiometer; the user can still adjust it in the conventional manner, and any changes made by the remote are instantly reflected in the position of the volume knob.

Things get trickier with interactions that are invisible—either because they use wireless communication or because they are invoked through learned gestures. However, we can still design the physical form of the object to encourage the right behaviour. Nothing inherent in RFID requires it to be laid out flat in a card, but this leads users towards the correct interaction of tapping their Oyster transport payment card onto the similarly flat reader surface when travelling on the London Underground.

Similar rules apply when designing physical interfaces. Don't overload familiar connectors with unfamiliar behaviours. For example, you shouldn't use 3.5mm audio jacks to provide power, although alternative "data-level"

uses are probably okay. And if you're designing a new connector completely, think about ways to prevent users from connecting it the wrong way round. littleBits (`http://littlebits.cc/about`) faced this problem when they were designing their modular, plug-together electronic circuit building blocks. They were looking for a way to make it easy and relatively foolproof to connect things together because the product is aimed at beginners. Their solution is a nice approach in this respect, using magnets to both discourage incorrect connection whilst also encouraging the correct connection.

SUMMARY

This chapter will have given you a deeper understanding of the emerging field of the Internet of Things and some ways to direct your thinking when you are designing something to fit into the landscape.

The examples have shown how you need to think not just about the technical details of how the device will work, but also of how it will fit into the wider context of the user's life. Unlike an app on her phone, your Internet of Things product will take up physical space in the world and won't be silenced just by the user's focusing on a different app, so you need to consider that for her.

You also need to take care not to divulge any information that users wouldn't expect you to. It is a new field full of expanding possibilities which gives us opportunities for delighting and enriching people's lives, but we need to do so in a way that doesn't scare or alienate the less technical in the population.

Careful use of touchpoints such as magic and fairytales can help with this, as will systems which fail gracefully and still perform their non-computer-enhanced functions in the way to which everyone is accustomed.

Even this early in the book you will already have encountered different aspects of the network bleeding through—for example, in the description of Natalie Jeremijenko's Live Wire or the mention of HTTP protocol versions in the section on graceful degradation.

That's hardly surprising in a book about the *Internet* of Things, but we don't assume that you have a full understanding of how the network works or exactly what it can do. The next chapter looks at the common protocols on the Internet (and even what a protocol is) and how they interrelate, to give you a better understanding of how the Internet works.